8K

УДК 004.89

М.А. Князева, В.А. Тимченко
Институт автоматики и процессов управления ДВО РАН, г. Владивосток, Россия
mak@imcs.dvgu.ru, rakot2k@mail.ru.

Преобразование программ из исходного представления в целевое представление на основе описаний проекций языка исходного представления на язык целевого представления*

В статье представлена общая идея использования проекционного подхода к решению задачи преобразования программ из заданного исходного представления в требуемое целевое представление. Приведен фрагмент модели онтологии проекций языков исходного представления программ на языки целевого представления и на примере продемонстрировано, как в соответствии с данной моделью представляется описание (фрагмента) конкретной проекции.

Задача преобразования исходного представления программ в целевое представление в соответствии с определенными правилами преобразования возникает везде, где необходимо проводить анализ, выполнять определенные преобразования программ и кодогенерацию. Актуальность ее определяется и тем, что с ней постоянно приходится сталкиваться при разработке компиляторов, в том числе оптимизирующих и распараллеливающих, а также систем, предназначенных для анализа, оптимизации и распараллеливания программ. Традиционно анализ и преобразования программ выполняются над программами, представленными не на языках программирования высокого уровня непосредственно, а при помощи различных схем и моделей, являющихся пригодным и удобным представлением для работы с исходными программами (например, схемы Мартынюка, схемы Лаврова, представления программ в виде различных графов) и подробно описаны, например, в работах [1-3]. Многие такие системы, в особенности те, которые работают с несколькими языками исходных текстов программ, имеют свое внутреннее промежуточное представление программ. При этом выполняется преобразование анализируемой программы на исходном языке (обычно это язык программирования высокого уровня – исходное представление) во внутреннее представление, используемое затем для последующей ее обработки (различные схемы, модели программы – целевое представление).

Решение данной задачи по-прежнему требует от разработчиков трансляторов и систем преобразования программ больших усилий ввиду ее трудоемкости. Чтобы сократить эти усилия, все чаще прибегают к помощи различных систем построения трансляторов, готовым компиляторам с готовыми грамматиками языков, реализованными сторонними разработчиками (системы построения трансляторов (СПТ) ANTLR, SableCC, Sage++, Bison, YACC, компиляторы Portland Group Inc.). Такой подход применен, например, при разработке систем OPC [4], Polaris [5], SUIF/SUIF2 [6]. Однако, несмотря на то, что в этом случае действительно экономятся усилия разработчиков, он имеет свои недостатки и ограничения [7], [8]. Использование внешних программ, библиотек,

* Работа выполнена при финансовой поддержке ДВО РАН, инициативный научный проект «Интернет-система управления информацией о преобразованиях программ» (06-III-A-01-007).

инструментальных средств автоматически ставит в зависимость от разработчиков этих средств. Кроме того, хотя и наблюдается тенденция к улучшению таких характеристик СПТ, как удобство и простота интерфейса транслятора, позволяющая его интегрировать в программное средство, удобство описания грамматики языка программирования, читаемость сгенерированного кода, тем не менее, при их использовании возникают определенные сложности.

В структурно-предикативной системе СПОРК [9], применен подход, основанный на реализации структурных предикативных грамматик и средств работы со структурным графом. Структурный граф является внутренним представлением программ, используемым в данной системе. Внутреннее представление является здесь результатом синтаксического и контекстного анализа программы с помощью структурно-предикативной грамматики (СП-грамматики).

Следует заметить, что СП-грамматика относится к классу логических грамматик, в которых описание правил ведется на языке первого порядка в терминах термов и логических формул. Оперирование таким языком может оказаться весьма затруд нительным для пользователей этой системы. Поэтому работы, направленные на сокращение усилий, затрачиваемых на преобразование программ из исходного представления в целевое продолжаются. И как следствие – актуальна разработка гибких, удобных методов формализации информации о языках представления программ, смягчающих требования к специализированной подготовке специалистов, использующих эти методы. Идеи, заложенные в проекционном подходе, позволяют разработать формальные модели, инкапсулирующие в себе информацию, необходимую для преобразования программ.

Проекционный подход к трансляции начал разрабатываться еще в начале 70-х гг. XX века в НИВЦ МГУ. Основные концепции данного подхода и выразительные средства для адекватного описания проекций были проработаны В.Ш. Кауфманом [10], [11]. Под *проекцией* языка L_1 на язык L_2 в его работах понимается отображение (в обычном математическом смысле) $\mathbf{p}$: $l_1 \rightarrow l_2$, где l_1 и l_2 – множества допустимых в этих языках текстов. Ценность проекции заключается в том, что она документирует способ интерпретации конструкций L_1 за счет средств L_2, а также выступает в роли важнейшей составной части задания на построение транслятора, оставляя свободу в выборе алгоритмов трансляции, поскольку не зависит ни от чего, кроме L_1 и L_2. Чтобы задать проекцию с языка L_1 на язык L_2, необходимо зафиксировать три соотношения. Соотношение, описывающее устройство текстов языка L_1, затем соотношение, описывающее устройство текстов языка L_2, и, наконец, соотношение, описывающее связь между этими двумя соотношениями.

Каждое соотношение представляет собой систему элементарных соотношений, которые выражаются в декларативном виде средствами V-языка [12].

Практическая ценность проекционного подхода как технологии создания тра нсляторов или интерпретаторов состоит, в частности, в разработке автоматизированной СПТ, которая должна решать проблему, как по формально заданной проекции (спецификации на транслятор) построить реализующий эту проекцию транслятор. Транслятор реализует проекцию, если для каждого текста на входном языке выдает в качестве результата образ этого текста в соответствии с проекцией. Это так называемая проблема разрешимости проекции. В рамках предложенного Кауфманом подхода она окончательно не была решена.

В настоящей работе предложен способ описания языковых проекций, ориентированный, с одной стороны, на доступность его освоения и применения специалистами, не обладающими высокой математической подготовкой, а с другой стороны, на реализуемость описанных проекций. Реализуемость означает принципиальную возможность

построения интерпретатора, который на основе описания проекции языка исходного представления на язык целевого представления и программы на языке исходного представления генерировал бы эту программу на языке целевого представления. Следует сразу оговориться, что множества языков исходного и целевого представлений ограничены процедурными языками программирования, а также языками моделей и схем, посредством которых моделируются программы на процедурных языках программирования. Таким образом, проблема разрешимости в данном случае не стоит, поскольку все описываемые проекции являются заведомо разрешимыми в силу способа их описания.

Подход к описанию языковых проекций. Основная идея

Идея применения проекционного подхода к решению задачи преобразования исходного представления программ в целевое представление на основе описания проекции языка исходного представления на язык целевого представления состоит в следующем.

Необходимо разработать и описать модель онтологии языка исходного представления, а также модель онтологии языка целевого представления.

Онтология языков исходного и целевого представления представляет собой совокупность информации, описывающую множество терминов языка, а также взаимосвязь между этими терминами, то есть способ их объединения в языковые конструкции. Можно сказать, что онтология языка определяет абстрактный синтаксис этого языка. Модель онтологии языка представлена семантической сетью терминов, связанных между собой направленными дугами (отношениями) и снабженной специальной разметкой.

Разработать и описать связь онтологии языков исходного и целевого представления с конкретным синтаксисом этих языков. Эта связь определяет содержание правильных конструкций языка с точки зрения синтаксиса этого языка. Связь онтологии языка с его конкретным синтаксисом содержит такие элементы языка, как пунктуационное наполнение, определяет порядок следования лексем и т.д. Данный вид информации ограничивает способ выражения в виде текста того смысла, который содержится в абстрактном представлении программы.

Модель онтологии языка представления программ и модель связи онтологии языка с конкретным синтаксисом этого языка необходимы для осуществления процедуры синтаксического анализа текстов программ на заданном языке и формирования представления этих программ в абстрактном синтаксисе, а также обратной процедуры – синтеза текстов программ по их представлению в абстрактном синтаксисе.

Программа в абстрактном синтаксисе – это программа, представленная в терминах онтологии языка программирования, не содержащая элементов конкретного синтаксиса.

И, наконец, необходимо разработать и описать модель онтологии проекций языков исходного представления на соответствующие им языки целевого представления, в соответствии с которой описывается проекция заданного исходного языка на заданный целевой язык. На основе данных моделей была разработана концепция подсистемы генерации программ в целевом представлении на основе описаний проекций языков исходного представления на языки целевого представления. Подсистема состоит из гибко настраиваемых на соответствующую модель онтологии программных компонент, а также управляемых соответствующими моделями онтологий, в которые также при необходимости могут вноситься изменения [7], [8].

Модель онтологии проекций языков исходного представления на языки целевого представления

Под проекцией будем понимать отображение следующего вида:

$$\mathbf{M}: E \to E',$$

где E – множество элементов языка исходного представления. Множество этих элементов, структурированных определенным образом, представляют собой онтологию языка, в терминах онтологии которого представляется исходная программа.

$$E = C \cup R \cup A,$$

где C – множество понятий (объектов) языка исходного представления;
R – множество отношений языка исходного представления;
A – множество атрибутов языка исходного представления.
E' – множество элементов языка целевого представления. Множество этих элементов, структурированных определенным образом, представляют собой онтологию языка, в терминах онтологии которого представляется целевая программа.

$$E' = C' \cup R' \cup A',$$

где C' – множество понятий (объектов) языка целевого представления;
R' – множество отношений языка целевого представления;
A' – множество атрибутов языка целевого представления.

Ниже представлен фрагмент модели онтологии проекций языков исходного представления программ на языки целевого представления. Данная модель представляет собой спецификацию абстрактного синтаксиса языка описания проекций. Описание операционной семантики языка опущено, оставлено только ее неформальное описание.

При описании использованы следующие обозначения языка спецификаций, который приведен в работе [13]: **объед** – объединение; [а] – а не обязательно (может отсутствовать); = – определение понятия; **:** – раскрытие позиции, * – возможно неопределенный атрибут; **сер** – серийная компонента.

Язык описания проекций = (**сер** описание проекции **:** Описание проекции)

Описание: Язык описания проекций определяет множество описаний проекций.

Семантика: Описание проекции = (язык исходного представления **:** СТРОК, язык целевого представления **:** СТРОК, соответствия **:** Соответствия)

Описание: В описании проекции определяется язык исходного представления программ, язык целевого представления программ и соответствия, которые описывают связь между ними.

Семантика: Соответствия = (**сер** соответствие **:** Описание соответствия)

Описание: Данный термин определяет множество описаний соответствий между элементами исходного и целевого языков представлений программ.

Семантика: Описание соответствия = (элемент исходного языка **:** СТРОК, конструкция целевого языка **:** Элементы целевого языка)

Описание: В соответствии описывается структура конструкции целевого языка, которая должна быть сопоставлена элементу исходного языка, то есть элементу исходного языка соответствует структурированный определенным образом набор элементов целевого языка.

Семантика: Для заданного элемента исходного языка интерпретировать описание структуры конструкции целевого языка, представленное в данном соответствии.

Элементы целевого языка = ([понятия **:** Набор понятия], [отношения **:** Набор отношений], [атрибуты **:** Набор атрибутов])

Описание: Конструкция целевого языка состоит из, возможно пустых, наборов понятий, отношений и атрибутов.

Семантика: Создать конструкцию целевого языка заданной структуры.

Набор понятий = (**сер** понятие целевого языка **:** Понятие)
Понятие = (имя **:** СТРОК)
Описание: *Набор понятий* представляет собой множество понятий целевого языка.

Каждое созданное понятие добавляется во множество, содержащее только те элементы, которые были созданы при интерпретации описания текущего соответствия. Перед интерпретацией следующего соответствия из этого множества должны быть удалены все элементы, то есть оно становится пустым.

Затем каждое созданное понятие добавляется в историю соответствий между элементом исходного языка и набором элементов целевого языка. В истории в процессе сканирования исходной программы запоминается информация о созданных элементах языка целевого представления, о том, какому элементу языка исходного представления они были сопоставлены. По существу по истории можно пошагово проследить процесс генерации программы на языке целевого представления.

Семантика: Последовательно создать понятия, множество имен которых определяет компонента *Понятие*. Для каждого созданного понятия:
добавить понятие во множество элементов, созданных в процессе интерпретации описания текущего соответствия;
добавить элемент исходного языка и понятие в историю соответствий между элементом исходного языка и набором элементов целевого языка.
Набор отношений = (**сер** отношение **:** Отношение)
Описание: *Набор отношений* представляет собой множество отношений.
Семантика: Отношение = (имя **:** СТРОК, понятие-начало отношения **:** Понятие, понятие-конец отношения **:** Понятие)
Описание: *Отношение* характеризуется своим именем и связывает два понятия – понятие, определяемое селектором «понятие-начало отношения», и понятие, определяемое селектором «понятие-конец отношения».
Семантика: Если понятие, определяемое селектором *понятие-конец отношения*, на момент интерпретации описания текущего соответствия создано, то

Создать отношение с именем, определяемым селектором *имя*, назначить понятие, определяемое селектором *понятие-начало отношения*, начальным понятием отношения, а понятие, определяемое селектором *понятие-конец отношения* – конечным понятием отношения.

Если понятие, определяемое селектором *понятие-конец отношения*, на момент интерпретации описания текущего соответствия еще не существует (не создано), то

Добавить информацию о данном отношении во множество несконструированных элементов. Это множество хранит информацию о тех элементах, описание структуры которых уже было получено, но они еще не могут быть созданы, поскольку не были созданы все необходимые элементы, составляющие структуру данного элемента.

Если при интерпретации очередного соответствия понятие, определяемое селектором *понятие-конец отношения*, появилось во множестве созданных в результате интерпретации данного соответствия элементов, то оно назначается конечным понятием данного отношения, после чего инициируется создание отношения. После создания отношения понятие удаляется из множества несконструированных элементов.

Набор атрибутов = (**сер** атрибут **:** Атрибут)
Описание: *Набор атрибутов* представляет собой множество атрибутов.
Семантика: Атрибут = (имя **:** СТРОК, аргумент **:** Аргумент атрибута, вычислимый **:** ЛОГ, значение **:** Значение атрибута)
Описание: *Атрибут* характеризуется своим именем, аргументом, на котором он определен, и значением, которое он может принимать. Кроме того, атрибут может быть вычислимым и невычислимым. Значение признака «вычислимый» указывает на то, откуда должно браться значение для атрибута: это будет либо значение терминального элемента сканируемой программы в абстрактном синтаксисе на исходном языке, либо значение, заданное непосредственно на этапе описания проекции.

Далее на основе указанной информации о типе значения атрибута возможны следующие способы означивания атрибута.

Если тип значения – идентификатор переменной, функции, константы или типа данных, то значение ищется среди элементов таблиц идентификаторов, констант, типов данных, присутствующих в программе и сформированных в результате лексико-синтаксического анализа программы. Если элемент с указанным значением найден, то он присваивается в качестве значения атрибуту, если не найден, то создается константа, переменная или тип данных, информация о нем заносится в соответствующую таблицу, после чего он присваивается в качестве значения атрибуту.

Если тип значения – понятие, то значение атрибута будет получено в результате поиска по элементам уже известного (сгенерированного) фрагмента программы на языке целевого представления.

Если тип значения – строка, целое число, вещественное число или логическое значение, то атрибуту присвоится значение, заданное на этапе описания проекции, либо значение первого найденного терминального элемента программы на исходном языке, которое будет извлечено в результате поиска по элементам программы на языке исходного представления.

Семантика: Создать атрибут с именем, определяемым селектором *имя*, аргумент которого определяется селектором *аргумент*, а в качестве значения присвоить ему компоненту *Значение атрибута*, определяемую селектором *значение*.

Если у данного атрибута значение, определяемое селектором *вычислимый – истина*, то значение атрибута, определяемое селектором *значение*, должно браться из программы в абстрактном синтаксисе на языке исходного представления, как значение первого найденного терминального элемента программы, и на этапе описания проекции его задавать не нужно. Если же значение атрибута будет задано на этапе описания проекции и указано, что атрибут вычислимый, то заданное значение будет проигнорировано.

Иначе, если значение, определяемое селектором *вычислимый – ложь*, то для атрибута берется значение, заданное на этапе описания проекции. Если значение атрибута не задано на этапе описания проекции и указано, что атрибут невычислимый, то считается, что его значение *пусто*.

Аргумент атрибута = (тип **:** Тип аргумента, значение **:** СТРОК)

Описание: *Аргумент атрибута* представляет собой пару (тип, значение). Тип аргумента атрибута, определяемый селектором *тип*, указывает на то, как интерпретировать собственно аргумент атрибута, определяемый селектором *значение*, а также определяет область допустимых значений для аргумента атрибута.

Семантика: Тип аргумента = **объед** (*Понятие*, *Идентификатор переменной*, *Идентификатор функции*, *Идентификатор константы*, *Идентификатор типа данных*)

Описание: *Тип аргумента* может быть одним из перечисленных.

Семантика: Значение атрибута = (тип **:** Тип значения, [значение **:** СТРОК])

Описание: *Значение атрибута* представляет собой пару (тип, значение). Тип значения атрибута, определяемый селектором *тип*, указывает на то, как интерпретировать значение атрибута, определяемое селектором *значение*, а также определяет область допустимых значений, которые может принимать атрибут.

Семантика: Тип значения = **объед** (*Строка*, *Целое число*, *Вещественное число*, *Логическое значение*, *Понятие*, *Идентификатор переменной*, *Идентификатор функции*, *Идентификатор константы*, *Идентификатор типа данных*)

Описание: *Тип значения* может быть одним из перечисленных.

Семантика: Зафиксируем языки исходного и целевого представлений программ и приведем фрагмент описания конкретной проекции. На рис. 1, рис. 2 и рис. 3 приведен фрагмент описания проекции языка программирования Паскаль (язык исходно го представления программ) на Язык моделей структурных программ (язык целевого представления программ) [14], [15], выполненный в соответствии с представленной

выше моделью онтологии. На рисунках представлены описания соответствий между понятиями языка программирования Паскаль «Выражение», «Оператор цикла While», «Сумма» и соответствующими им конструкциями языка моделей структурных программ.



Рисунок 1 – Фрагмент описания проекции языка программирования Паскаль на Язык моделей структурных программ. Описание соответствия для понятия «Выражение»

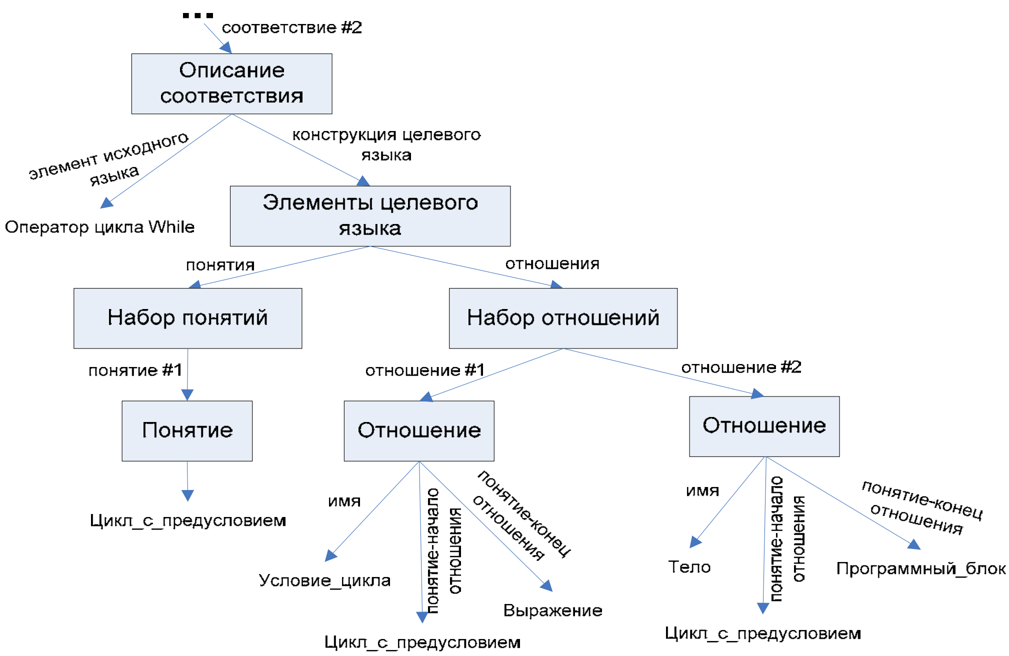


Рисунок 2 – Фрагмент описания проекции языка программирования Паскаль на Язык моделей структурных программ (продолжение). Описание соответствия для понятия «Оператор цикла While»

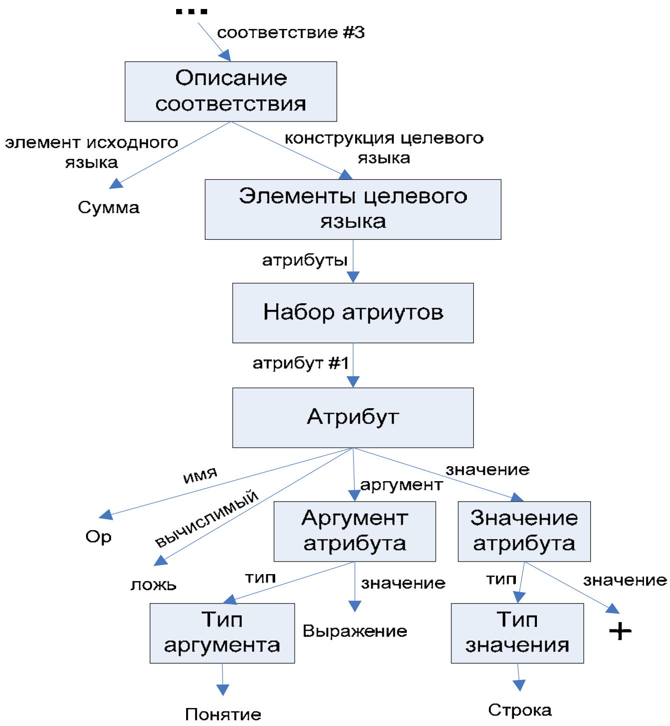


Рисунок 3 – Фрагмент описания проекции языка программирования Паскаль на Язык моделей структурных программ (продолжение). Описание соответствия для понятия «Сумма»

Заключение

В статье показано, что задача преобразования программ из некоторого исходного представления в требуемое целевое по-прежнему актуальна. Она вызывает трудности, несмотря на уже, казалось бы, хорошо проработанные, постоянно совершенствующиеся подходы к ее решению. Последние, несмотря на то, что все ближе подходят к их преодолению, все же имеют свои недостатки и ограничения. В работе описана общая идея использования проекционного подхода для решения задачи преобразования программ из исходного представления в целевое представление на основе описания проекции языка исходного представления на язык целевого представления, который позволяет преодолеть недостатки, присутствующие в существующих подходах. Приведен фрагмент модели онтологии проекций языков исходного представления программ на языки целевого представления и продемонстрировано, как в соответствии с моделью онтологии представляется описание (фрагмента) конкретной проекции.

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ – Петербург, 2002.
2. Воеводин Вл.В. Теория и практика исследования параллелизма последовательных программ // Программирование. – 1992. – № 3. – С. 38-54.
3. Касьянов В.Н. Оптимизирующие преобразования программ. – М.: Наука, 1988.
4. Штейнберг Б.Я. Открытая распараллеливающая система // Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. – Ростов-н/Д.: Изд-во Рост. ун-та, 2004. – С. 166-182.

5. Blume W., Doallo R., Eigenmann R. a.o. Parallel programming with Polaris // Computer. – 1992. – Vol. 29, № 12. – P. 78-82.
6. Wilson R.P., French R.S., Wilson C.S. a.o. SUIF: An infrastructure for research on parallelizing and optimizing compilers // SIGPLAN Not. – 1994. – Vol. 29, № 12. – P. 31-37.
7. Князева М.А., Тимченко В.А. Подсистема генерации единого внутреннего представления в системе преобразований программ // Программные продукты и системы. – 2008. – № 1. – С. 58-62.
8. Клещев А.С., Князева М.А., Тимченко В.А. Интеллектуальная система генерации единого внутреннего представления в системе преобразований программ // Труды II Международной конференции «Системный анализ и информационные технологии»: В 2 т. – Т. 1. – М.: Изд-во ЛКИ, 2007. – 288 с.
9. Тапкинов Б.Ю. Внутреннее представление программы в Системе построения оптимизирующих и распараллеливающих компиляторов // Труды Всероссийской научной конференции «Научный сервис в сети Интернет: технологии параллельного программирования», Новороссийск, 18 – 23 сентября, 2006. – С. 88-91.
10. Кауфман В.Ш. О технологии создания трансляторов (проекционный подход). // Программирование. – 1978. – № 5. – С. 36-44.
11. Бунимова Э.О., Кауфман В.Ш., Левин В.А. Об описании языковых проекций // Вестн. Моск. ун-та. Вычисл. матем. и киберн. – 1978. – № 4. – С. 68-73.
12. Кауфман В.Ш., Левин В.А. Естественный подход к проблеме описания контекстных условий // Вестн. Моск. ун-та. Вычисл. матем. и киберн. – 1977. – № 2. – С. 67-77.
13. Ершов А.П., Грушецкий В.В. Метод описания алгоритмических языков, ориентированный на реализацию: Препр. / ВЦ Сибирское отделение АН СССР. – Новосибирск, 1977.
14. Артемьева И.Л., Князева М.А., Купневич О.А. Модель онтологии предметной области «Оптимизация последовательных программ». Ч. 1: Термины для описания объекта оптимизации // НТИ. Сер. 2. – 2002. – № 12. – С. 23-28.
15. Артемьева И.Л., Князева М.А., Купневич О.А. Модель онтологии предметной области «Оптимизация последовательных программ». Ч. 2: Термины для описания процесса оптимизации // НТИ. Сер. 2. – 2003. – № 1. – С. 22-29.

М.О. Князєва, В.А. Тимченко

Перетворення програм з початкового представлення у цільове представлення на основі описів проекцій мови початкового представлення на мову цільового представлення

У статті представлена загальна ідея використання проекційного підходу до рішення задачі перетворення програм із заданого початкового представлення в необхідне цільове представлення. Наведений фрагмент моделі онтології проекцій мов початкового представлення програм на мови цільового представлення і на прикладі продемонстровано, як відповідно до даної моделі представляється опис (фрагмента) конкретної проекції.

M.A. Knyazeva, V.A. Timchenko

Transformation of Programs from the Source Representation to Target Representation Based on the Ontology Model for Mappings of Programs' Source Representation Language into Programs' Target Representation Language

The paper presents the general idea of using the mapping approach for solving the task of transformation of programs from the source representation to required target representation. The paper demonstrates a fragment of the ontology model for mappings of programs' source representation languages into programs' target representation languages and gives the example of how a description of (a fragment of) a particular mapping is represented in accordance with the ontology model.

Статья поступила в редакцию 09.07.2008.